Investigatory Project for AISSCE - 2020-21 Computer Science (083) Practical Exam

Student Information

NAME: DIPTANIL SAHA
CLASS: 12 A
BOARDS ROLL NO: 12645615
ADMISSION NUMBER: 1728/10
TOPIC: PARKING RECORD MAINTAINING SYSTEM USING FLASK (PYTHON MICRO WEB FRAMEWORK) AND MYSQL (CODE NAME: PARKWITHFLASK)
SUBJECT TEACHER: MR. ARIJIT GHOSH
SCHOOL: KENDRIYA VIDYALAYA COSSIPORE

Topic of Project

Parking Record Maintaining System using Flask (Python Micro web framework) and MySQL
Short Name: ParkwithFlask

Group Members

1. Diptanil Saha
2. Ankit Singh
3. Rajarshi Pal

SCAN THIS FOR LIVE DEMO

Acknowledgement

I would like to express my special gratitude to our Principal Shri K.K. Dubey, our teacher cum guide Mr. Arijit Ghosh and other teachers of the school who gave me guidance to complete this project. Also, I am thankful to the various internet sites and my school library from where I could get instant help whenever I got stuck in the course of completing my project.

I consider myself lucky to have such a wonderful batch of classmates whose motivation and help made it possible for me to complete this project within the given time frame.

Certificate

This is to certify that this investigatory project work in the subject of Computer Science has been done by Diptanil Saha of class 12 - A in the academic year 2020-21 for AISSCE Practical Examination conducted by CBSE on Parking Record Maintaining System using Flask (Python Micro web framework) and MySQL (ParkwithFlask). The student has successfully completed the project in Computer Science under the guidance of Mr. Arijit Ghosh as per the syllabus prescribed by CBSE.

Signature of Internal Examiner

Signature of External Examiner

Principal's Signature

Content

Introduction

The project is based on managing parking areas for malls or any other parking areas. This system has been developed using Flask (Python Micro Web Framework) and MySQL. This system can log the cars into the parking facility and logout the cars from the parking facility with the provision of calculating the parking charge on per hour basis. Moreover, ParkwithFlask has provision of multi-level user access i.e. admin level, entry level and exit level. Admin level has the ability to register new employees to log the cars into the facility or to logout the cars out of the facility which is separated out for security purposes. Admin level also has the ability to register new employees/users to log the cars into their facility and update some details like parking charge etc.

Python

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

MySQL

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. A relational database organizes data into one or more data tables in which data types may be related to each other; these relations help structure the data. SQL is a language programmer use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an RDBMS like MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups.

Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Applications that use the Flask framework include Pinterest and LinkedIn.

Features of Flask

- Development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja templating
- Support for secure cookies (client-side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Extensive documentation
- Google App Engine compatibility
- Extensions available to enhance features desired

Minimum System Requirements

Operating System:	Windows, Mac and Linux
CPU:	x86 64-bit CPU (Intel / AMD architecture)
RAM:	1 GB
Software:	Chrome, Python IDLE, MySQL CLI
• **Frontend:** • **Backend:**	• HTML, CSS, Javascript • Python, MySQL
Hard Drive:	512 MB

Project Layout

```
parkwithflask
├──app
│   ├──static
│   │   ├──android-chrome-192x192.png
│   │   ├──android-chrome-512x512.png
│   │   ├──apple-touch-icon.png
│   │   ├──browserconfig.xml
│   │   ├──favicon-16x16.png
│   │   ├──favicon-32x32.png
│   │   ├──favicon.ico
│   │   ├──mstile-70x70.png
│   │   ├──mstile-144x144.png
│   │   ├──mstile-150x150.png
│   │   ├──mstile-310x150.png
│   │   ├──mstile-310x310.png
│   │   ├──safari-pinned-tab.svg
│   │   └──site.webmanifest
│   ├──templates
│   │   ├──403.html
│   │   ├──404.html
│   │   ├──base.html
│   │   ├──entry.html
│   │   ├──exit-car.html
│   │   ├──exit.html
│   │   ├──forget-password.html
│   │   ├──history.html
│   │   ├──index.html
│   │   ├──login.html
│   │   ├──manage-parking.html
│   │   ├──manage-user.html
│   │   ├──print.html
│   │   ├──register-successful.html
│   │   ├──register.html
│   │   └──update-user.html
│   ├──__init__.py
│   ├──forms.py
│   ├──models.py
│   └──routes.py
├──migrations/*
├──config.py
├──parkwithflask.py
└──requirements.txt
```

app: This folder contains python files, static files for websites and templates for frontend purpose.
migration: This folder is created by system on migrating the database.
config.py: This python script contains all sorts of configuration for the project like MySQL url, email configuration and other configurations.
parkwithflask.py: This file coordinates with the files at app and helps to execute the project.
requirements.txt: This txt file contains all the python packages used in the project.

Python Code

- **app/__init__.py**

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
from flask_login import LoginManager
from flask_principal import Principal
from config import Config
from flask_mail import Mail, Message
app = Flask(__name__)
app.config.from_object(Config)
db = SQLAlchemy(app)
migrate = Migrate(app, db)
login = LoginManager(app)
principals = Principal(app)
mail = Mail(app)
db.init_app(app)
from app import routes, models
```

- **app/forms.py**

```
from flask_wtf import FlaskForm
from wtforms import SelectField, StringField, PasswordField, BooleanField,
SubmitField, FloatField
from wtforms.validators import ValidationError, DataRequired, Email, EqualTo,
Regexp, Length
from app.models import User, ParkingSlot, ParkingPrice, ParkingHistory
class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired()])
    remember_me = BooleanField('Remember Me')
    submit = SubmitField('Sign In')
class RegistrationForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    name = StringField('Name', validators=[DataRequired(), Regexp('[a-zA-Z]')])
```

```
    email = StringField('Email', validators=[DataRequired(), Email()])
    role = SelectField('Role', choices=[(' ', 'Select your
choice'),('Entry','ENTRY'),('Exit','EXIT')], validators=[DataRequired()])
    submit = SubmitField('Register')

    def validate_username(self, username):
        user = User.query.filter_by(username=username.data).first()
        if user is not None:
            raise ValidationError('Please use a different username.')
    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user is not None:
            raise ValidationError('Please use a different email address.')
    def validate_role(self, role):
        if role==" ":
            raise ValidationError('Please select a role!')

class EntryForm(FlaskForm):
    registration_no = StringField('Registration', validators=[DataRequired(),
Regexp('[0-9A-Z]', message='Enter Correct Registration Number!')])
    name = StringField('Name', validators=[DataRequired(), Regexp('[a-zA-Z]',
message='Enter only Name!')])
    phone = StringField('Phone Number', validators=[DataRequired(), Regexp('[0-
9]', message='Enter only Number!')])
    submit = SubmitField('Entry')

    def validate_registration_no(self, registration_no):
        rc = ParkingSlot.query.filter_by(rc=registration_no.data).first()
        if rc is not None:
            raise ValidationError('This Registration Number is already logged into
your Parking Facility!')
    def validate_phone(self, phone):
        phone = ParkingSlot.query.filter_by(phone=phone.data).first()
        if phone is not None:
            raise ValidationError('This Phone number is already registered into
your Parking Facility!')

class ExitForm(FlaskForm):
    phone = StringField('Phone Number', validators=[DataRequired()])
    submit = SubmitField('Validate Exit!')
    def validate_phone(self, phone):
        phone = ParkingSlot.query.filter_by(phone=phone.data).first()
        if phone is None:
            raise ValidationError('Phone Number didn\'t matched!')

class ForgetPassword(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    submit = SubmitField('Request New Password')
```

```
    def validate_email(self, email):
        user = User.query.filter_by(email = email.data).first()
        if user is None:
            raise ValidationError('Email Address is not Registered!')
class UpdateUser(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email()])
    submit = SubmitField('Register')
    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user is not None:
            raise ValidationError('Please use a different email address.')
class UpdatePrice(FlaskForm):
    charge = FloatField('Parking Charge/hr', validators=[DataRequired()])
    submit = SubmitField('Update')
```

- **app/models.py**

```
from flask import current_app
from datetime import datetime
from app import db, login
from flask_login import UserMixin
from werkzeug.security import generate_password_hash, check_password_hash
from random import randint

class User(UserMixin, db.Model):
    __tablename__ = 'user'
    id = db.Column(db.Integer, primary_key = True)
    username = db.Column(db.String(64), index = True, unique= True, nullable =
False)
    email = db.Column(db.String(120), index=True, unique=True)
    name = db.Column(db.String(64), index = True, nullable = False)
    password_hash = db.Column(db.String(128))
    role = db.Column(db.String(10), nullable=False, index=True)
    def __repr__(self):
        return '<User %r>' % self.id
    def set_password(self, password):
        self.password_hash = generate_password_hash(password)
    def check_password(self, password):
        return check_password_hash(self.password_hash, password)
@login.user_loader
def load_user(id):
    return User.query.get(int(id))
class ParkingSlot(db.Model):
    __tablename__ = 'parkingslot'
    id = db.Column(db.Integer, primary_key=True)
    rc = db.Column(db.String(10), index = True, unique = True, nullable=False)
    phone = db.Column(db.String(10), index = True, unique = True, nullable=False)
    name = db.Column(db.String(20), index = True, nullable=False)
```

```
    entry_empname = db.Column(db.String(64), index = True, nullable=False)
    entry_time = db.Column(db.DateTime, index = True, nullable=False)
    entry_epoch = db.Column(db.Integer, index = True, nullable=False)
    def __repr__(self):
        return '<ParkingSlot %r>' % self.id
class ParkingHistory(db.Model):
    __tablename__ = 'parkinghistory'
    id = db.Column(db.Integer, primary_key= True)
    rc = db.Column(db.String(10), index = True, nullable = False)
    phone = db.Column(db.String(10), index = True, nullable = False)
    name = db.Column(db.String(20), index = True, nullable = False)
    entry_empname = db.Column(db.String(64), index = True, nullable = False)
    entry_time = db.Column(db.DateTime, nullable = False, index = True)
    exit_empname = db.Column(db.String(64), index = True, nullable = False)
    exit_time = db.Column(db.DateTime, nullable = False, index = True)
    time_stayed = db.Column(db.String(15), nullable=False, index=True)
    parking_charge = db.Column(db.Float, index=True, nullable = False)
    def __repr__(self):
        return '<ParkingHistory %r>' % self.id
class ParkingPrice(db.Model):
    __tablename__ = 'parkingprice'
    id = db.Column(db.Integer, primary_key = True)
    date_updated = db.Column(db.DateTime, nullable = False, index = True)
    charge = db.Column(db.Float, nullable = False)
    def __repr__(self):
        return '<ParkingPrice %r>' % self.id
```

- **app/routes.py**

```
import secrets
import string
from flask import render_template, flash, redirect, url_for, request, current_app,
abort, session, jsonify
from flask_login import login_user, logout_user, current_user, login_required
from werkzeug.urls import url_parse
from app import app, db, mail, login
from app.forms import LoginForm, RegistrationForm, EntryForm, ExitForm,
ForgetPassword, UpdateUser, UpdatePrice
from app.models import User, ParkingSlot, ParkingPrice, ParkingHistory
from flask_principal import Permission, RoleNeed, Identity, AnonymousIdentity,
identity_changed, identity_loaded
from flask_mail import Message
from datetime import datetime, timedelta, date
from time import time
@login.unauthorized_handler
def unauthorized_callback():
    return redirect('/login?next=' + request.path)
@identity_loaded.connect_via(app)
```

```
def on_identity_loaded(sender, identity):
    # Set the identity user object
    identity.user = current_user
    # Assuming the User model has a list of roles, update the
    # identity with the roles that the user provides
    if hasattr(current_user, 'role'):
        # for role in current_user.role:
        identity.provides.add(RoleNeed(str(current_user.role)))
admin_permission = Permission(RoleNeed('Admin'))
entry_permission = Permission(RoleNeed('Entry'))
exit_permission = Permission(RoleNeed('Exit'))
admin_entry_permission = admin_permission.union(entry_permission)
admin_exit_permission = admin_permission.union(exit_permission)
all_permission = admin_entry_permission.union(admin_exit_permission)
if not User.query.filter(User.email ==
'diptanilsahakvckolkata@gmail.com').first():
    user = User(username = 'ADMN001', email = 'diptanilsahakvckolkata@gmail.com',
name = 'Diptanil Saha', role = 'Admin')
    user.set_password('ParkwithFlask')
    db.session.add(user)
    db.session.commit()
parking_price = ParkingPrice.query.order_by(ParkingPrice.id).first()
if parking_price is None:
    parking_price = ParkingPrice(date_updated = datetime.now(), charge = 50.0)
    db.session.add(parking_price)
    db.session.commit()
@app.route('/')
@app.route('/index')
def index():
    return render_template('index.html', title='Home')
@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('index'))
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        if user is None or not user.check_password(form.password.data):
            flash('Invalid username or password')
            return redirect(url_for('login'))
        login_user(user, remember=form.remember_me.data)
        identity_changed.send(current_app._get_current_object(),
identity=Identity(user.id))
        next_page = request.args.get('next')
        if not next_page or url_parse(next_page).netloc != '':
            next_page = url_for('index')
        return redirect(next_page)
    return render_template('login.html', title='Sign In', form=form)
```

```
@app.route('/logout')
@login_required
def logout():
    logout_user()
    for key in ('identity.name', 'identity.auth_type'):
        session.pop(key, None)
identity_changed.send(current_app._get_current_object(),identity=AnonymousIdentity
())
    return redirect(url_for('index'))
@app.route('/register', methods=['GET', 'POST'])
@login_required
def register():
    if admin_permission:
        with admin_permission.require(http_exception=403):
            form = RegistrationForm()
            if form.validate_on_submit():
                user = User(username=form.username.data,name=form.name.data,
email=form.email.data, role=form.role.data)
                password = ''.join(secrets.choice(string.ascii_lowercase +
string.digits) for i in range(8))
                user.set_password(str(password))
                db.session.add(user)
                db.session.commit()
                msg = Message('Password for your account with ParkwithFlask',
                sender='noreply.parkwithflask@gmail.com',
                recipients= [form.email.data])
                msg.body = """
                Hello {name},
                You are successfully registered on ParkwithFlask.
                Your credentials are:
                Username: {username}
                Password: {password}
                Role: {role}
                If you face any problem, contact your Portal admin.
                Thank You,
                Team ParkwithFlask
                """.format(name = form.name.data, role = form.role.data, username
= form.username.data, password = str(password))
                mail.send(msg)
                return render_template('register-successful.html',
title=str(form.username.data + ' registered successfully'), form=form)
            return render_template('register.html', title='Register', form=form)
    else:
        flash('Hmm, seems that you\'re not authorized to visit /register!')
        return redirect(url_for('index'))
@app.route('/entry', methods=['GET', 'POST'])
@login_required
def entry():
```

```
    if admin_entry_permission:
        with admin_entry_permission.require(http_exception=403):
            form = EntryForm()
            if form.validate_on_submit():
                avail_space = ParkingSlot.query.order_by(ParkingSlot.id).all()
                count = 0
                for i in avail_space:
                    count+=1
                if count <= 100:
                    entry = ParkingSlot(rc = form.registration_no.data, phone =
form.phone.data, name = form.name.data,
                                        entry_empname = current_user.name,
entry_time = datetime.fromtimestamp(int(time())), entry_epoch = int(time()))
                    db.session.add(entry)
                    db.session.commit()
                    flash(str(form.registration_no.data+' entered successfully!'))
                    return redirect('/entry')
                else:
                    flash('No space available!')
                    return redirect('/entry')
            return render_template('entry.html', title='Entry', form=form)
    else:
        flash('Hmm, seems that you\'re not authorised to visit /entry!')
        return redirect(url_for('index'))
@app.route('/exit')
@login_required
def exit():
    if admin_exit_permission:
        with admin_exit_permission.require(http_exception=403):
            exitData = ParkingSlot.query.order_by(ParkingSlot.id).all()
            return render_template('exit.html', title = 'Exit', exitData=exitData)
    else:
        flash('Hmm, seems that you\'re not authorised to visit /exit!')
        return redirect(url_for('index'))

@app.route('/exit/<int:id>', methods=['GET','POST'])
@login_required
def exit_delete(id):
    if admin_exit_permission:
        with admin_exit_permission.require(http_exception=403):
            get_car = ParkingSlot.query.get_or_404(id)
            form = ExitForm()
            ptime = int(time())-get_car.entry_epoch
            get_charge =
ParkingPrice.query.order_by(ParkingPrice.date_updated).first()
            charge = get_charge.charge
            hour = get_hour(ptime)
            fare = charge * hour
```

```
            exit_time = int(time())
            time_stayed = str(timedelta(seconds=(exit_time-get_car.entry_epoch)))
            phone = str('******'+str(get_car.phone)[6:10])
            if form.validate_on_submit():
                upload_history = ParkingHistory(
                rc = get_car.rc, phone = get_car.phone, name = get_car.name,
entry_empname=  get_car.entry_empname, entry_time = get_car.entry_time, exit_time
= datetime.fromtimestamp (exit_time),exit_empname = current_user.name,time_stayed
= time_stayed,parking_charge = fare)
                db.session.add(upload_history)
                db.session.delete(get_car)
                db.session.commit()
                flash(str(get_car.rc + ' has been removed from your parking
facility!'))
                return redirect('/exit')
            return render_template('exit-car.html', title = str(get_car.rc + ' -
Exit'), car=get_car, form=form, fare=fare, phone=phone, time=time_stayed,
exit_time=datetime.fromtimestamp(exit_time))
    else:
        flash('Hmm, seems that you\'re not authorised to visit /exit!')
        return redirect(url_for('index'))
@app.route('/history')
@login_required
def history():
    if all_permission:
        with all_permission.require(http_exception=403):
            get_history =
ParkingHistory.query.order_by(ParkingHistory.exit_time).all()
            return render_template('history.html', title = 'History', history =
get_history)
    else:
        return 'You are on History page!'

@app.route('/forget-password', methods=['GET', 'POST'])
def forget_password():
    form = ForgetPassword()
    if form.validate_on_submit():
        user = User.query.filter_by(email = form.email.data).first()
        password = ''.join(secrets.choice(string.ascii_lowercase + string.digits)
for i in range(8))
        user.set_password(str(password))
        db.session.commit()
        msg = Message('Password for your account with ParkwithFlask',
            sender='noreply.parkwithflask@gmail.com',
            recipients= [form.email.data])
        msg.body = """
            Hello {name},
            You have requested change password. Your revised credentials are
```

```
            Username: {username}
            Password: {password}
            If you face any problem, contact your Portal admin.
            Thank You,
            Team ParkwithFlask
            """.format(name = user.name, role = user.role, username =
user.username, password = str(password))
        mail.send(msg)
        flash('Password Successfully Changed! Check your email!')
        return redirect('/')
    return render_template('forget-password.html', title = 'Forget Password',
form=form)
@app.route('/manage-user')
@login_required
def manage():
    if admin_permission:
        with admin_permission.require(http_exception=403):
            all_user = User.query.order_by(User.id).all()
            return render_template('manage-user.html', title='Manage-User',
all_user=all_user)
    else:
        flash('Hmm, trying to sneek into the manage user! You\'re not allowed!')
        return redirect(url_for('index'))
@app.route('/manage-user/update/<int:id>', methods = ['GET', 'POST'])
@login_required
def manage_update(id):
    if admin_permission:
        with admin_permission.require(http_exception=403):
            user = User.query.filter_by(id = id).first()
            form = UpdateUser()
            if form.validate_on_submit():
                user.email = form.email.data
                db.session.commit()
                flash('Data Successfully Updated!')
                return redirect('/manage-user')
            return render_template('update-user.html', form=form, user=user)
    else:
        flash('Hmm, trying to sneek into the manage user! You\'re not allowed!')
        return redirect(url_for('index'))
@app.route('/manage-user/delete/<int:id>')
@login_required
def manage_delete(id):
    if admin_permission:
        with admin_permission.require(http_exception=403):
            user = User.query.filter_by(id = id).first()
            if user.role != "Admin":
                db.session.delete(user)
                db.session.commit()
```

```
                flash('User Successfully Deleted!')
                return redirect('/manage-user')
            else:
                flash('Error!')
                return redirect('/manage-user')
    else:
        flash('Hmm, trying to sneek into the manage user! You\'re not allowed!')
        return redirect(url_for('index'))
@app.route('/manage-parking', methods = ['GET', 'POST'])
@login_required
def manage_parking():
    if admin_permission:
        with admin_permission.require(http_exception=403):
            parkinghistory =
ParkingHistory.query.order_by(ParkingHistory.id).all()
            income = 0
            for data in parkinghistory:
                income = income + data.parking_charge
            parking_price =
ParkingPrice.query.order_by(ParkingPrice.date_updated).first()
            date_updated = parking_price.date_updated
            charge = parking_price.charge
            form = UpdatePrice()
            if form.validate_on_submit():
                parking_price.charge = form.charge.data
                parking_price.date_updated = datetime.now()
                db.session.commit()
                flash('Charges Successfully Updated!')
                return redirect('/manage-parking')
            return render_template('manage-parking.html', form=form, title='Manage
Parking', income=income, date=date_updated, charge = charge)
    else:
        flash('Hmm, trying to sneek into the manage parking facility page! You\'re
not allowed!')
        return redirect(url_for('index'))
@app.route('/available-space')
@admin_entry_permission.require(http_exception=403)
def available_space():
    avail_space = ParkingSlot.query.order_by(ParkingSlot.id).all()
    count = 0
    for i in avail_space:
        count+=1
    return str(100-count)
@app.route('/about')
def about():
    return 'Currently working on it!'
def get_hour(seconds):
    hour = seconds//3600
```

```
    seconds %= 3600
    minute = seconds // 60
    seconds %= 60
    if hour == 0:
        hour += 1
    else:
        if minute < 60 and minute > 0:
            hour+=1
        else:
            if seconds < 60 and seconds > 0:
                minute +=1
                if minute < 60 and minute > 0:
                    hour+=1
    return hour
@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html', title='Page Not Found!'),404
@app.errorhandler(403)
def page_not_found(e):
    return render_template('403.html', title='Unauthorised'),403
```

- **config.py**

```
import os
basedir = os.path.abspath(os.path.dirname(__file__))
class Config(object):
    SECRET_KEY = os.environ.get('SECRET_KEY')
    SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL') or \
     'mysql+pymysql://parkwithflask:password@localhost/parkwithflask'
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    MAIL_SERVER = 'smtp.gmail.com'
    MAIL_PORT = 465
    MAIL_USERNAME = 'noreply.parkwithflask@gmail.com'
    MAIL_PASSWORD = 'password'
    MAIL_USE_TLS = False
    MAIL_USE_SSL = True
    JSON_SORT_KEYS = False
```

- **parkwithflask.py**

```
from app import app, db
from app.models import User, ParkingSlot, ParkingPrice, ParkingHistory
@app.shell_context_processor
def make_shell_context():
    return {'db': db, 'User': User, 'ParkingSlot':ParkingSlot,
'ParkingHistory':ParkingHistory, 'ParkingPrice':ParkingPrice}
if __name__ == "__main__":
    app.run()
```

HTML CODE:-

For HTML code refer to this link-
https://geekedblue.pythonanywhere.com/view-files

MySQL

Database: parkwithflask
Tables: parkingslot, parkinghistory, parkingprice and user

parkingslot

```
mysql> desc parkingslot;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| id            | int         | NO   | PRI | NULL    | auto_increment |
| rc            | varchar(10) | NO   | UNI | NULL    |                |
| phone         | varchar(10) | NO   | UNI | NULL    |                |
| name          | varchar(20) | NO   | MUL | NULL    |                |
| entry_empname | varchar(64) | NO   | MUL | NULL    |                |
| entry_time    | datetime    | NO   | MUL | NULL    |                |
| entry_epoch   | int         | NO   | MUL | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

parkinghistory

```
mysql> desc parkinghistory;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| id             | int         | NO   | PRI | NULL    | auto_increment |
| rc             | varchar(10) | NO   | MUL | NULL    |                |
| phone          | varchar(10) | NO   | MUL | NULL    |                |
| name           | varchar(20) | NO   | MUL | NULL    |                |
| entry_empname  | varchar(64) | NO   | MUL | NULL    |                |
| entry_time     | datetime    | NO   | MUL | NULL    |                |
| exit_empname   | varchar(64) | NO   | MUL | NULL    |                |
| exit_time      | datetime    | NO   | MUL | NULL    |                |
| time_stayed    | varchar(20) | NO   | MUL | NULL    |                |
| parking_charge | float       | NO   | MUL | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
10 rows in set (0.96 sec)
```

parkingprice

```
mysql> desc parkingprice;
+--------------+----------+------+-----+---------+----------------+
| Field        | Type     | Null | Key | Default | Extra          |
+--------------+----------+------+-----+---------+----------------+
| id           | int      | NO   | PRI | NULL    | auto_increment |
| date_updated | datetime | NO   | MUL | NULL    |                |
| charge       | float    | NO   |     | NULL    |                |
+--------------+----------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

user

```
mysql> desc user;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id            | int          | NO   | PRI | NULL    | auto_increment |
| username      | varchar(64)  | NO   | UNI | NULL    |                |
| email         | varchar(120) | YES  | UNI | NULL    |                |
| name          | varchar(64)  | NO   | MUL | NULL    |                |
| password_hash | varchar(128) | YES  |     | NULL    |                |
| role          | varchar(10)  | NO   | MUL | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

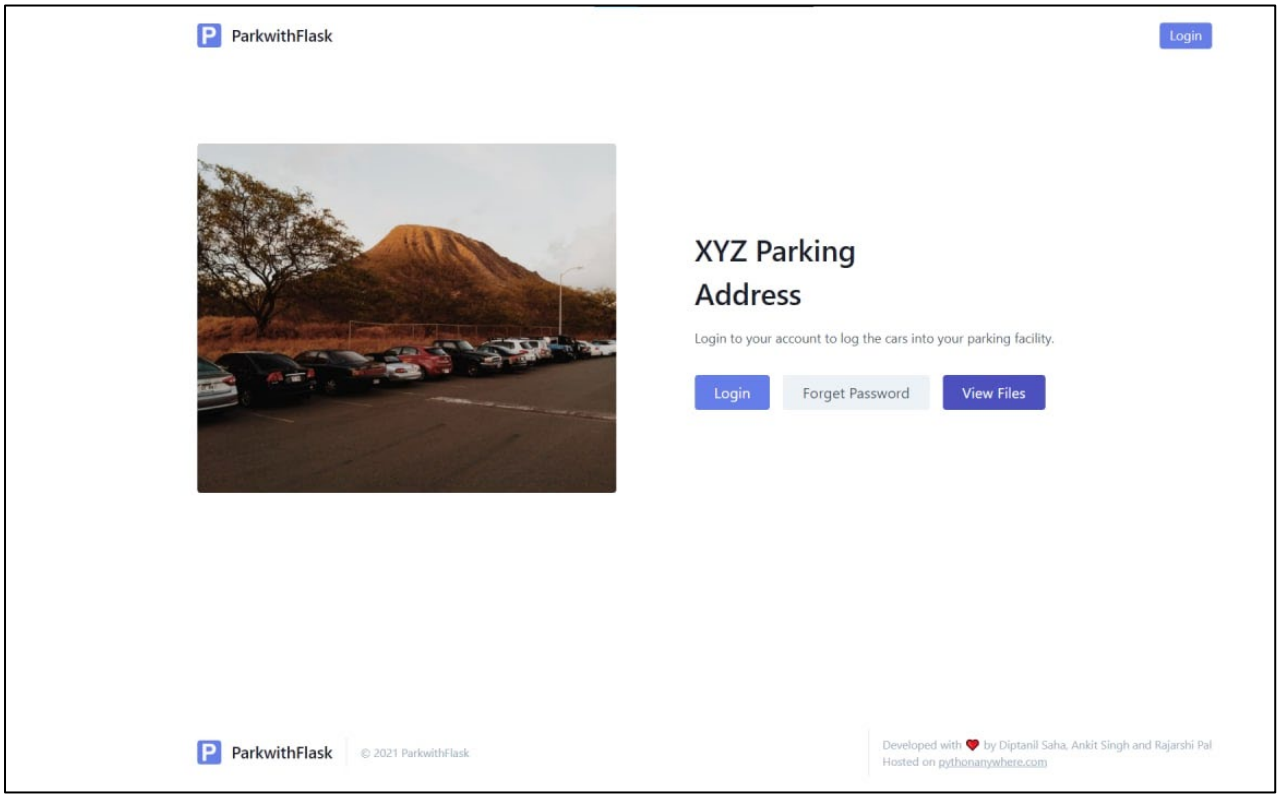
MySQL Dump

```
-- MySQL dump 10.13  Distrib 8.0.19, for Win64 (x86_64)
--
-- Host: localhost    Database: parkwithflask
-- ------------------------------------------------------
-- Server version 8.0.19
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
--
-- Table structure for table `alembic_version`
--
DROP TABLE IF EXISTS `alembic_version`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `alembic_version` (
  `version_num` varchar(32) COLLATE utf8_bin NOT NULL,
```

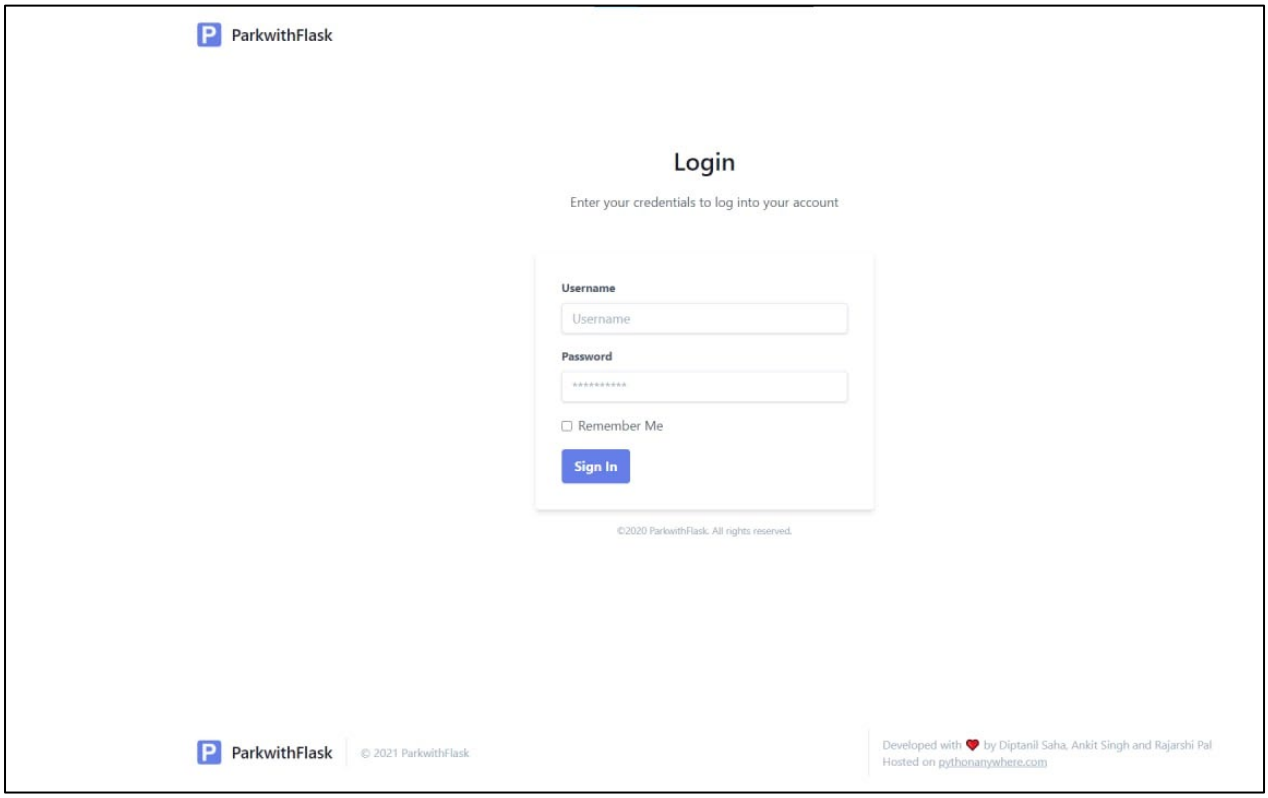
```
  PRIMARY KEY (`version_num`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Dumping data for table `alembic_version`
--
LOCK TABLES `alembic_version` WRITE;
/*!40000 ALTER TABLE `alembic_version` DISABLE KEYS */;
INSERT INTO `alembic_version` VALUES ('462bf544d36b');
/*!40000 ALTER TABLE `alembic_version` ENABLE KEYS */;
UNLOCK TABLES;
--
-- Table structure for table `parkinghistory`
--
DROP TABLE IF EXISTS `parkinghistory`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `parkinghistory` (
  `id` int NOT NULL AUTO_INCREMENT,
  `rc` varchar(10) COLLATE utf8_bin NOT NULL,
  `phone` varchar(10) COLLATE utf8_bin NOT NULL,
  `name` varchar(20) COLLATE utf8_bin NOT NULL,
  `entry_empname` varchar(64) COLLATE utf8_bin NOT NULL,
  `entry_time` datetime NOT NULL,
  `exit_empname` varchar(64) COLLATE utf8_bin NOT NULL,
  `exit_time` datetime NOT NULL,
  `time_stayed` varchar(20) COLLATE utf8_bin NOT NULL,
  `parking_charge` float NOT NULL,
  PRIMARY KEY (`id`),
  KEY `ix_parkinghistory_entry_empname` (`entry_empname`),
  KEY `ix_parkinghistory_entry_time` (`entry_time`),
  KEY `ix_parkinghistory_exit_empname` (`exit_empname`),
  KEY `ix_parkinghistory_exit_time` (`exit_time`),
  KEY `ix_parkinghistory_name` (`name`),
  KEY `ix_parkinghistory_parking_charge` (`parking_charge`),
  KEY `ix_parkinghistory_phone` (`phone`),
  KEY `ix_parkinghistory_rc` (`rc`),
  KEY `ix_parkinghistory_time_stayed` (`time_stayed`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Dumping data for table `parkinghistory`
--
LOCK TABLES `parkinghistory` WRITE;
/*!40000 ALTER TABLE `parkinghistory` DISABLE KEYS */;
INSERT INTO `parkinghistory` VALUES (9,'WB02A1234','9876543201','Ram','Diptanil
Saha','2020-12-16 12:26:16','Diptanil Saha','2020-12-16 12:26:24','0:00:08',50);
/*!40000 ALTER TABLE `parkinghistory` ENABLE KEYS */;
UNLOCK TABLES;
--
-- Table structure for table `parkingprice`
--
DROP TABLE IF EXISTS `parkingprice`;
```

```
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `parkingprice` (
  `id` int NOT NULL AUTO_INCREMENT,
  `date_updated` datetime NOT NULL,
  `charge` float NOT NULL,
  PRIMARY KEY (`id`),
  KEY `ix_parkingprice_date_updated` (`date_updated`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Dumping data for table `parkingprice`
  UNIQUE KEY `ix_user_email` (`email`),
  KEY `ix_user_name` (`name`),
  KEY `ix_user_role` (`role`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Dumping data for table `user`
--
LOCK TABLES `user` WRITE;
/*!40000 ALTER TABLE `user` DISABLE KEYS */;
INSERT INTO `user` VALUES (1,'ADMN001','diptanilsahakvckolkata@gmail.com','Diptanil
Saha','pbkdf2:sha256:150000$HIQEzNvL$42d4bea1f5747b74e95231f9a19536084d060f7e6e63a70da82cd4
6c7c282ac6','Admin');
/*!40000 ALTER TABLE `user` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
-- Dump completed on 2020-12-24 10:54:49
```

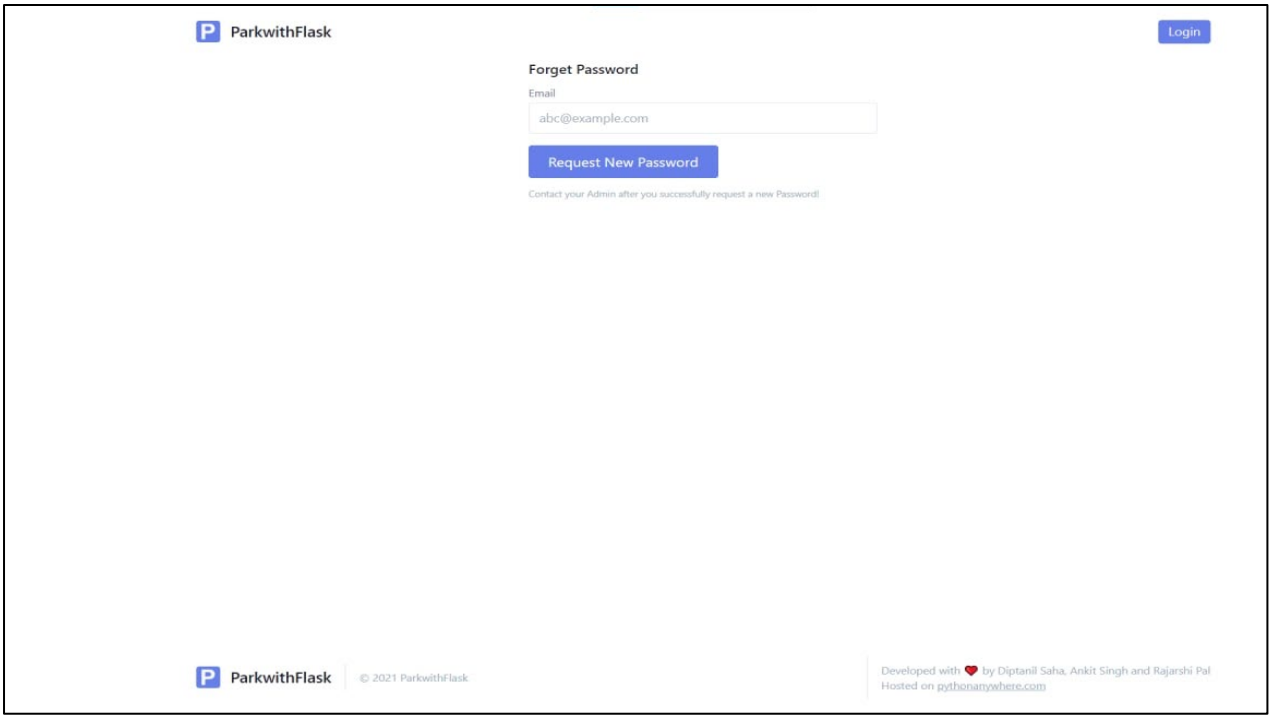
Screenshot



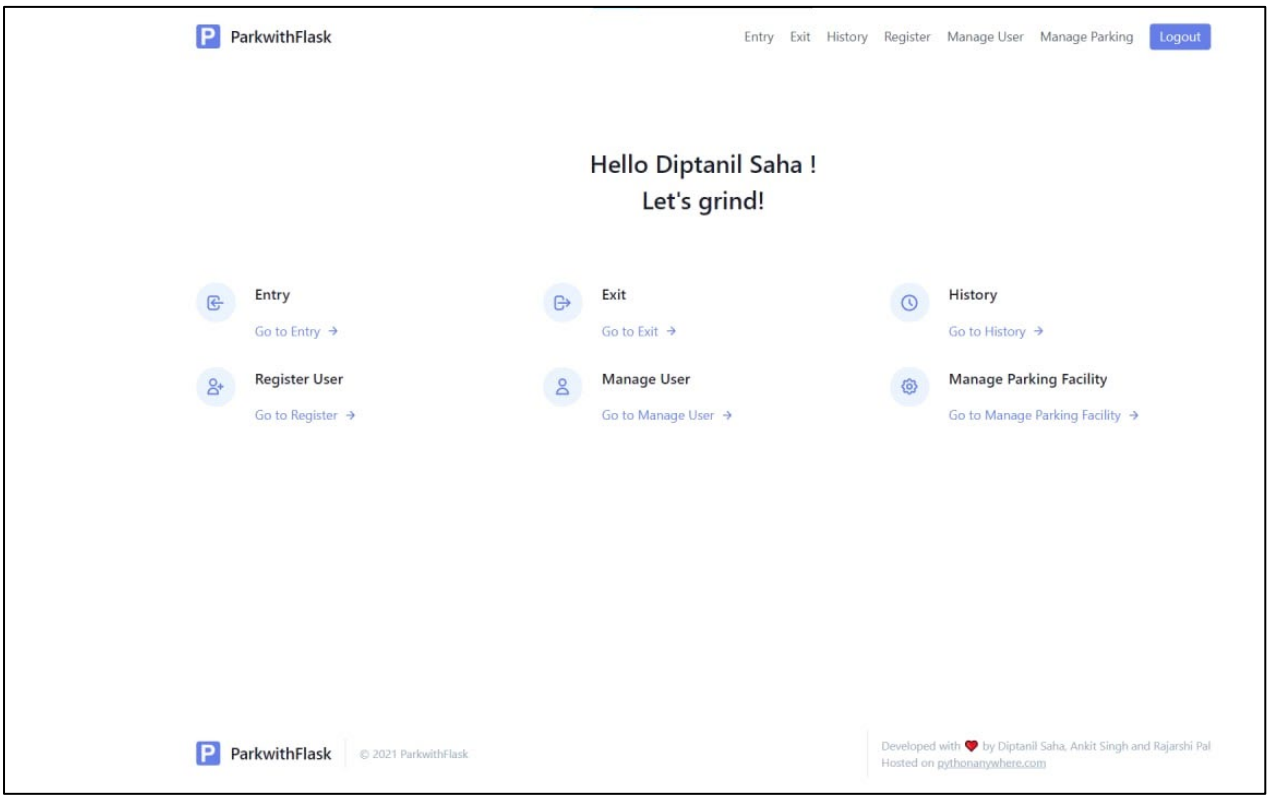
Landing Page



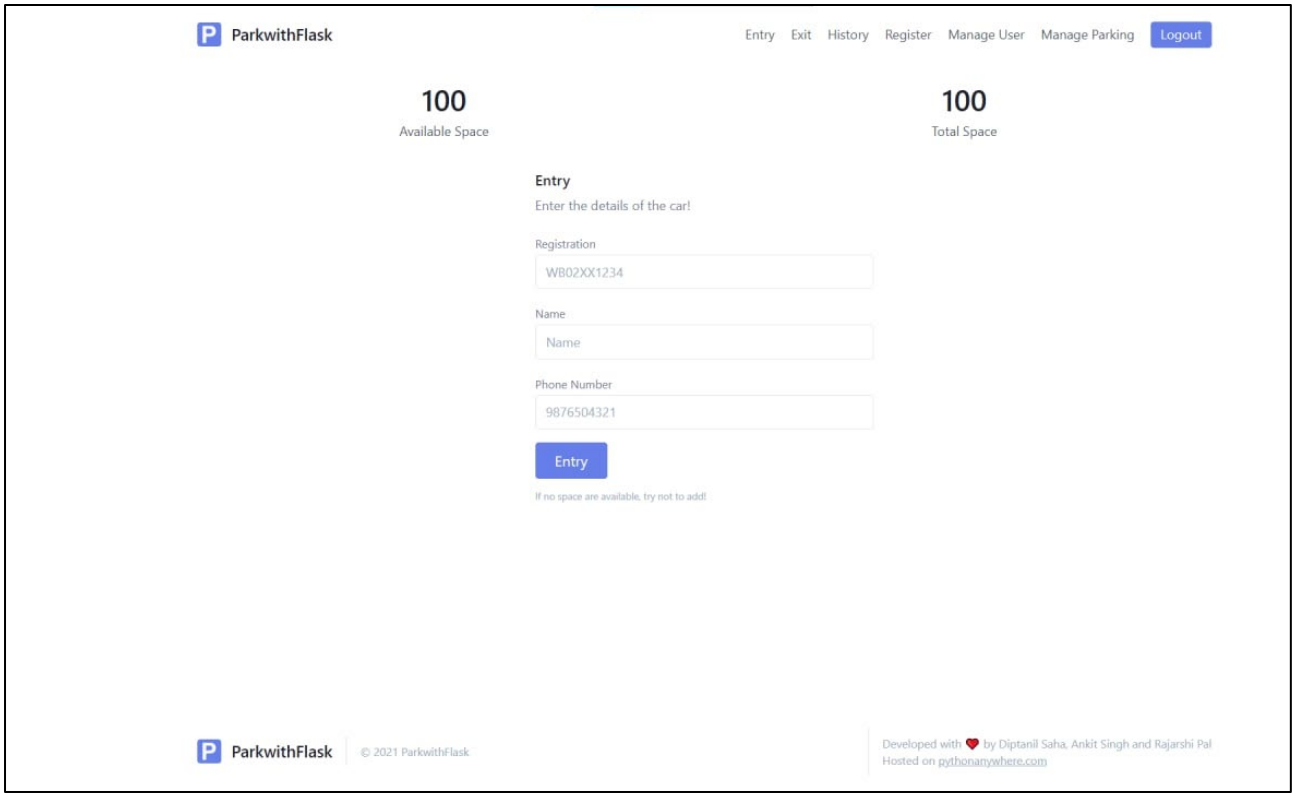
Login Page



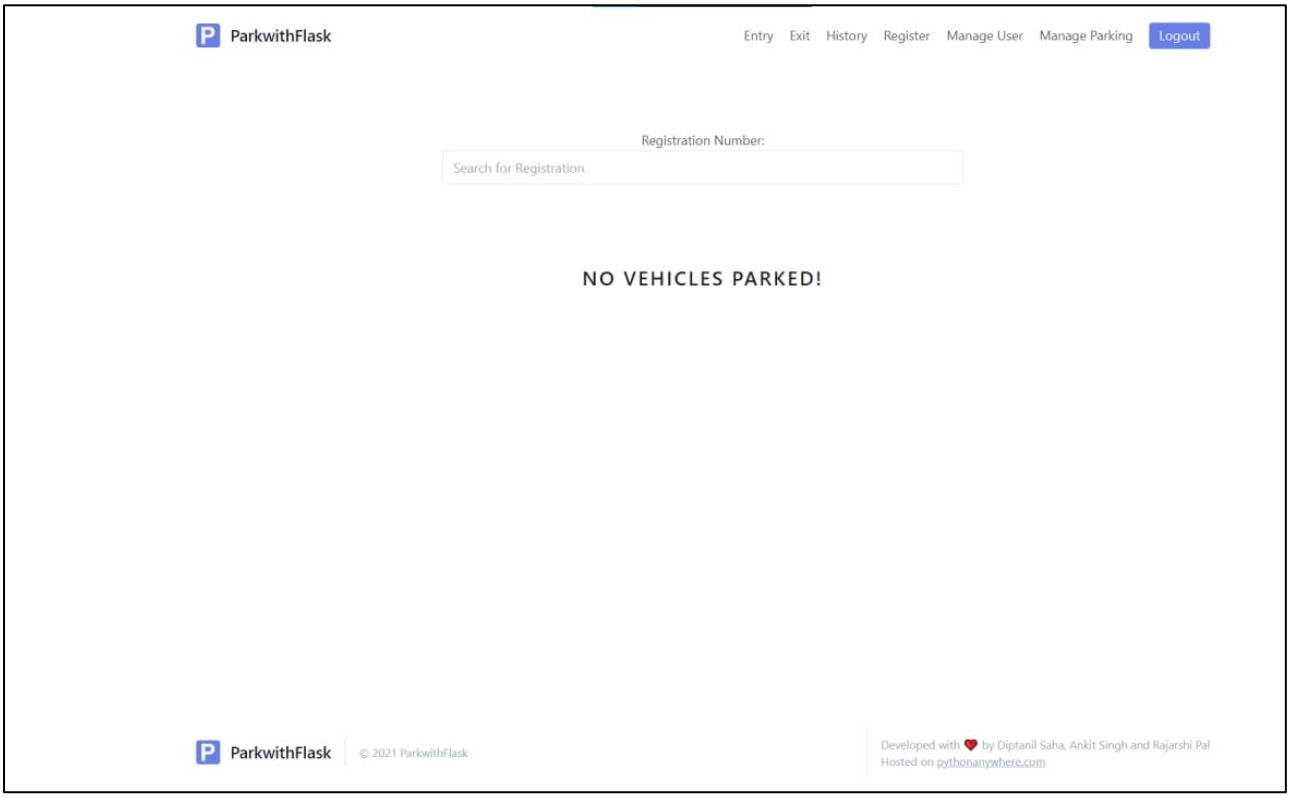
Forget Password



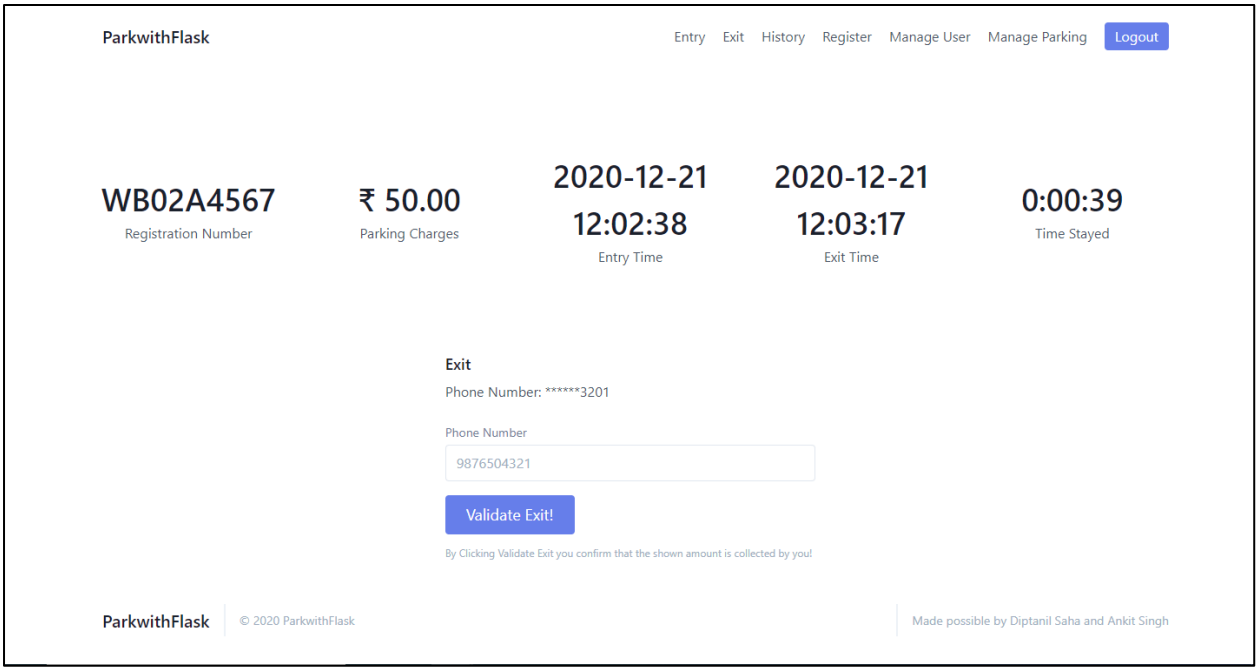
Admin Landing Page



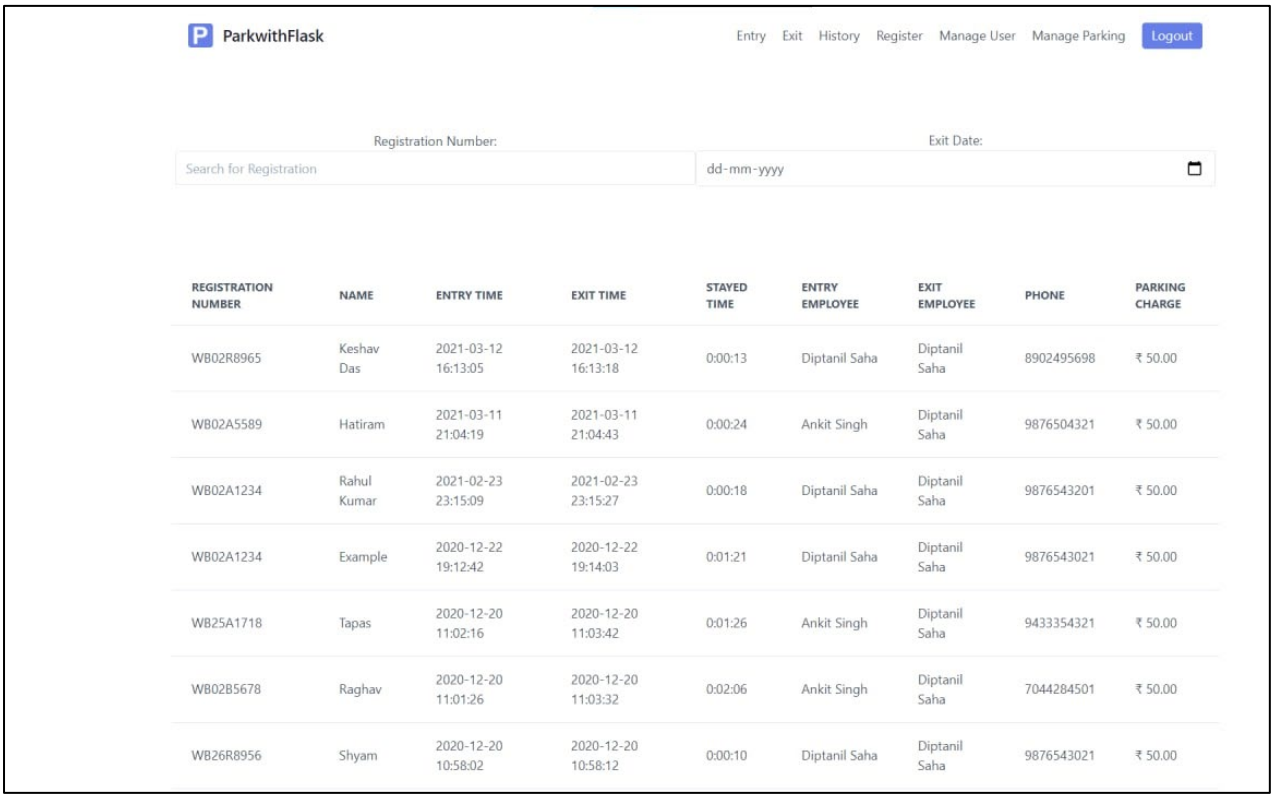
Entry Page



Exit Page

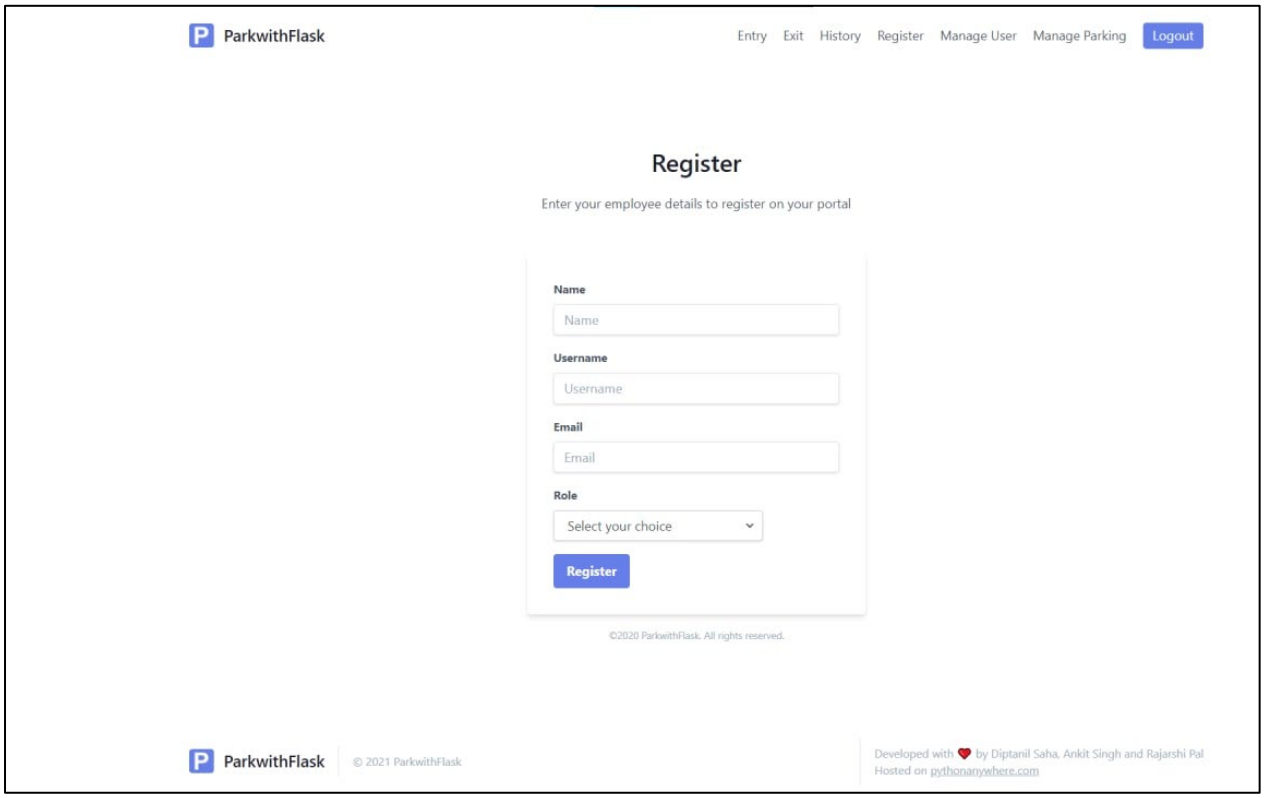


Exit For The Given Page

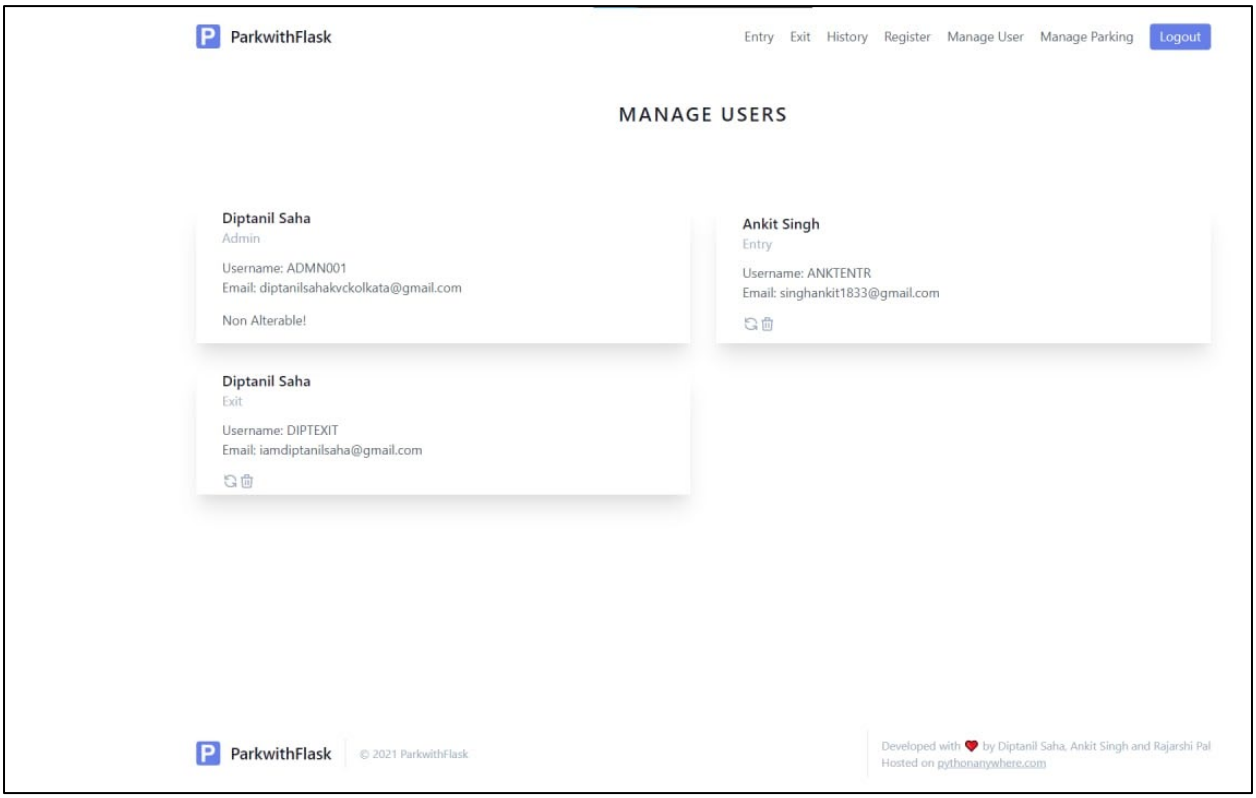


REGISTRATION NUMBER	NAME	ENTRY TIME	EXIT TIME	STAYED TIME	ENTRY EMPLOYEE	EXIT EMPLOYEE	PHONE	PARKING CHARGE
WB02R8965	Keshav Das	2021-03-12 16:13:05	2021-03-12 16:13:18	0:00:13	Diptanil Saha	Diptanil Saha	8902495698	₹ 50.00
WB02A5589	Hatiram	2021-03-11 21:04:19	2021-03-11 21:04:43	0:00:24	Ankit Singh	Diptanil Saha	9876504321	₹ 50.00
WB02A1234	Rahul Kumar	2021-02-23 23:15:09	2021-02-23 23:15:27	0:00:18	Diptanil Saha	Diptanil Saha	9876543201	₹ 50.00
WB02A1234	Example	2020-12-22 19:12:42	2020-12-22 19:14:03	0:01:21	Diptanil Saha	Diptanil Saha	9876543021	₹ 50.00
WB25A1718	Tapas	2020-12-20 11:02:16	2020-12-20 11:03:42	0:01:26	Ankit Singh	Diptanil Saha	9433354321	₹ 50.00
WB02B5678	Raghav	2020-12-20 11:01:26	2020-12-20 11:03:32	0:02:06	Ankit Singh	Diptanil Saha	7044284501	₹ 50.00
WB26R8956	Shyam	2020-12-20 10:58:02	2020-12-20 10:58:12	0:00:10	Diptanil Saha	Diptanil Saha	9876543021	₹ 50.00

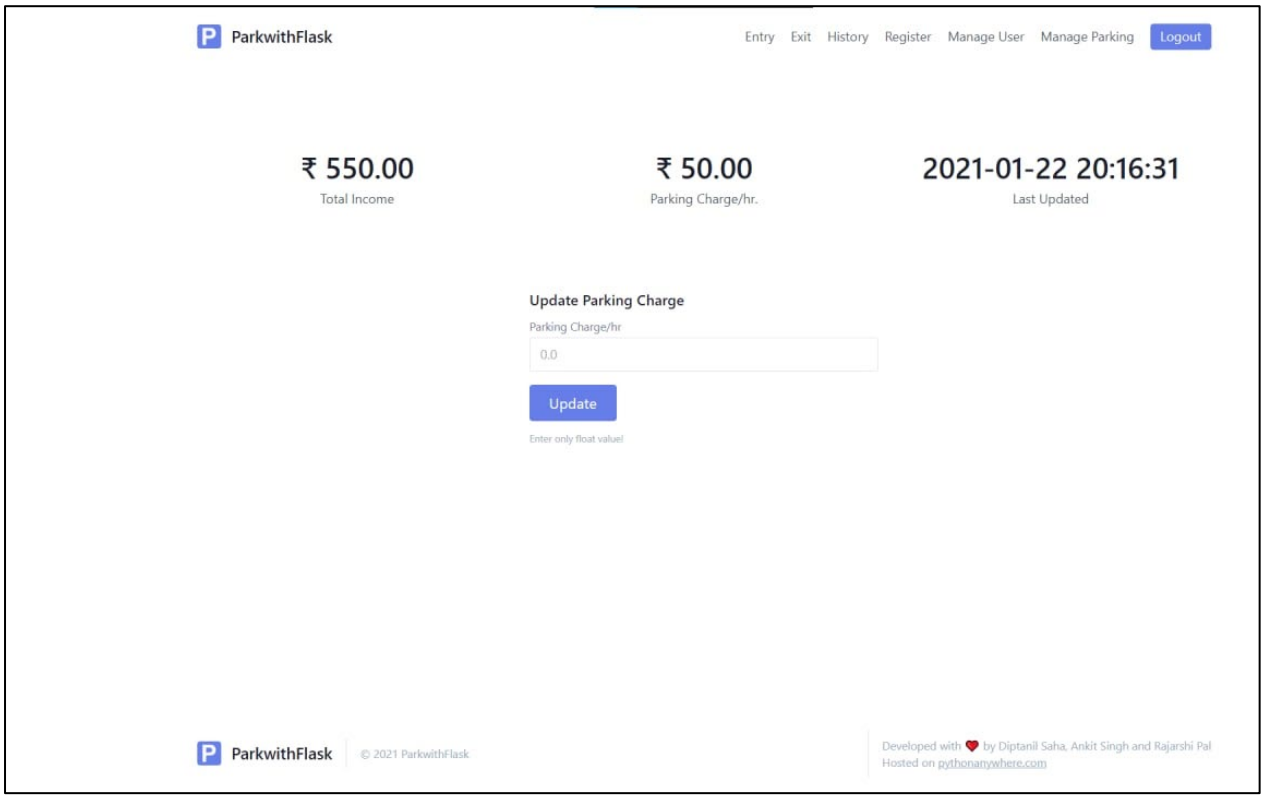
History Page



Register Page



Manage Users Page



Manage Parking Page

Features of the Project

- Multiple Role Based Access User
- Separate users for entry and exit
- Realtime update car login and car logout
- Multiple Role Based User Interface
- Entry and exit of car using details of the car driver

Future Scope of the Project

- This project is only for a single organization. However, this project can be upgraded for multiple organizations usage from a single database. Moreover, phone number-based authentication can be upgraded to two factor authentications by sending OTP via SMS for logging out cars from the parking facility.
- In future OpenCV can be used for autonomous parking system.

Bibliography

Books

- Sumita Arora, ***Computer Science with Python Textbook for Class XI and Class XII,*** Dhanpat Rai
- Grinberg, Miguel. ***Flask Web Development: Developing Web Applications with Python***. 2 ed., Shroff/O'Reilly, 2014. *Flask Web Development, 2nd Edition*, https://www.oreilly.com/library/view/flask-web-development/9781491991725/.
- Grinberg, M. (2014). ***The New and Improved Flask Mega-Tutorial*** https://courses.miguelgrinberg.com/p/flask-mega-tutorial

Website

- https://www.google.com
- https://stackoverflow.com/search?q=flask&s=ee50dd59-5669-4efc-b493-3c423b5bc84d
- https://flask.palletsprojects.com/en/1.1.x/
- https://flask-login.readthedocs.io/en/latest/
- https://pythonhosted.org/Flask-Mail/
- https://flask-migrate.readthedocs.io/en/latest/
- https://pythonhosted.org/Flask-Principal/
- https://pythonhosted.org/Flask-Security/
- https://flask-wtf.readthedocs.io/en/stable/